Volume 4, Issue 12, December 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Improved Approach for Text Retrieval in Distributed Environment

Anil Kumar*
Department of CSE
NITTTR, Chandigrah, India

Amit Doegar
Department of CSE
NITTTR, Chandigrah, India

Arvind Kumar Tiwari
Department of CSE
IIT (BHU), Varanasi, India

Abstract: Now days the basic problem is to retrieve the relevant documents from the distributed databases. Database selection and document selection are the basic problem to retrieve the relevant document in distributed environment. This paper presents two algorithms to retrieve the n most relevant documents across multiple databases for a given query 'q'. The SYNCH algorithm collect those databases from number of databases which contain our query 'q' and then find the score of each database and then select most appropriate database for the given query 'q'. After databases selection MIN algorithms retrieve 'n' most relevant documents from the selected databases and deleted those document which is similar to the document of other databases. This approach reduces the searching time and enhances the precision and recall rate. This proposed approach will be robust, effective and more efficient among the existing methods.

Keywords: Meta search engine; database selection; document selection, Crawler, multilevel search engine

I. INTRODUCTION

The Internet has become a vast information source in recent years. To help ordinary users find desired data in the Internet, many search engines[3] have been created. Each search engine has a corresponding database that defines the set of documents that can be searched by the search engine. Usually, an index for all documents in the database is created and stored in the search engine. For each term which represents a content word or a combination of several content words, this index can identify the documents that contain the term quickly. The pre-existence of this is critical for the search engine to answer user queries efficiently. Two types or search engines exist. General-purpose search engines attempt to provide searching capabilities for all documents in the Internet or on the Web. WebCrawler, HotBot, Lycos and Alta Vista are a few of such well-known search engines. Special-purpose search engines focus on documents in confined domains such as documents in an organization or of a specific interest. Large number of special-purpose search engines is currently running in the Internet.

A more practical approach to providing search services to the entire Internet[13] is the multi-level approach. At the bottom level are the local search engines. These search engines can be grouped based on the relatedness of their database, to form next level search engines (called metasearch engines). Lower, level metasearch engines can themselves be grouped to form higher level metasearch engines. This process can be repeated until there is only one metasearch engine at the top. A metasearch engine [3] is essentially an interface and it does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of the (meta) search engines at a lower level to provide better service. When a metasearch engine receives a user query, it first passes to the appropriate (meta) search engines at the next level recursively until real search engines are encountered, and then collects recognizes the results from real search engines, possible going through metasearch engines at lower levels. A two-level search engine organization is illustrated in Figure1. The advantages of this approach are as follows

(a) User queries can (eventually) be evaluated against smaller databases in parallel, resulting in reduced response time.
(b) updates to indexes can be localized, i.e., the index of a local search engine is updated only when documents in its database are modified. Local updates may need to be propagated to upper level metadata that represent the contents of local databases; the propagation can be done infrequently as the metadata are typically statistical in nature and can tolerate certain degree of inaccuracy.
(c) Local information can be gathered more easily and in a timelier manner;
(d) The demand on storage space and processing power at each local search engine is more manageable. In other words, many problems associated with employing a single super search engine can be overcome or greatly alleviated when this multi-level approach is used.

When the number of search engines invokable by a metasearch engine is large, a serious inefficiency may arise. Typically, for a given query, only a small fraction of all search engines may contain useful documents to the query. As a result, if every search engine is blindly invoked for each user query, then substantial unnecessary network traffic will be created when the query is sent to useless search engines. In addition, local resources will be wasted when useless database are searched.

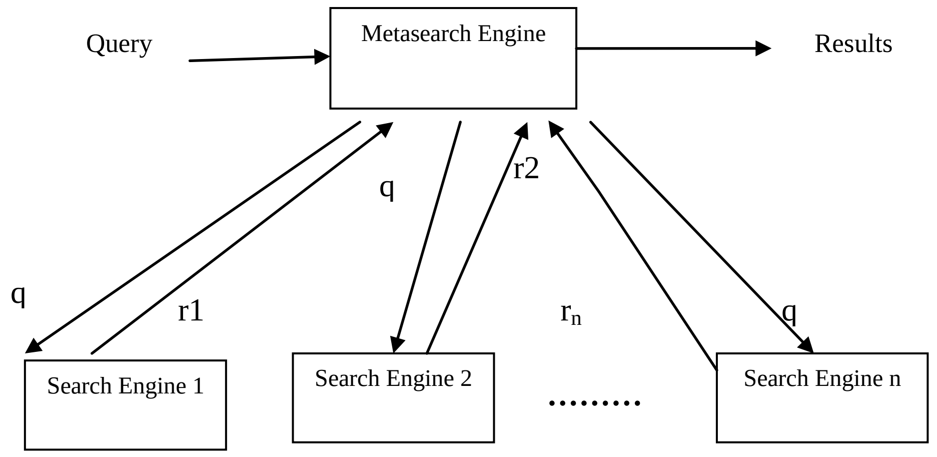


Figure 1: Two-Level Search Engine Organization

A better approach is to first identify those search engines that are most likely to provide useful results to a given query and then pass the query to only these search engines for desired documents. A challenging problem with this approach is how to identify potentially useful search engines. The current solution to this problem is to rank[7] all underlying databases in decreasing order of usefulness for each query using some metadata that describe the contents of each database. Often, the ranking is based on some measure which ordinary users may not be able to utilize to fit their needs. For a given query, the current approach can tell the user, to some degree of accuracy, which search engine is likely to be the most useful, the second most useful, etc. While such a ranking can be helpful, it cannot tell the user about the usefulness of any particular search engine.

II. PRIOR APPROACH

The solutions to the database selection problem can be roughly classified into the following four categories.

1. The naive approach does not perform database selection and the meta- search engine simply sends each user query to all search engines. This approach is employed by Meta Crawler [5].
2. Qualitative Approaches use rather detailed information such as the document frequency of each term to represent the contents of a database. Based on the information, a ranking score of each database can be computed to reflect the relative usefulness (or quality) of a database to a query. However, the score of a database is not equivalent to the number of potentially useful documents in the database. Qualitative approach is employed by a version of gGlOSS [7], CORI net [9] and D-WISE [7].
3. Quantitative Approaches predict the usefulness of each database based on measures that are much easier to understand than the measures used in qualitative approaches. In other words, the measures used in quantitative approaches reflect the usefulness of a database with respect to a given query more directly and explicitly in comparison to qualitative approaches. Usually, one or more pieces of statistical information need to be kept in order to enable the database usefulness prediction in quantitative approaches. One measure used by quantitative approaches is the “the number of documents in a database whose similarity with a given query is above a threshold” [12], [4]. The usefulness of a database to a query is defined to be the number of documents in the database that have high potentials to be useful to the query, that is, the similarities between the query and the documents as measured by a certain global similarity function are higher than a specified threshold. This usefulness can be defined precisely as follows:
Usefulness (T, q, D) = |{d|d € D and sim(q, d) > T}|
Where T is a threshold, D is a database, sim (q, d) is the similarity (closeness) between a query q and a document d in D. A query is simply a set of words submitted by a user. In practice, users may not know how to relate thresholds to the number of documents they like to retrieve. Therefore, users are more likely to tell the system the number of most similar documents (to their query) they like to retrieve directly. Such a number can be translated into a threshold by computing the usefulness of each database in decreasing thresholds.
4. Learning-based Approaches make use of past retrieval experiences with respect to a database to determine the usefulness of the database. Savvy- Search [10] is an example of a learning-based approach.

The proposed solutions to the document selection problem can also be classified into seven categories. The local determination approach simply allows each local search engine to return all documents it retrieved.

1. The user determination approach lets the user determine how many documents should be retrieved from each local search engine. Meta Crawler and Savvy Search use this approach.
2. The weighted allocation approaches retrieve proportionally more documents from local search engines whose databases have higher ranking scores. CORI net and D-WISE employ such approaches.
3. The OptDocRetrv algorithm retrieves documents from the databases, after the databases have been ranked.
4. The High-Correlation method provides useful services in large distributed information systems for document selection.

5. Learning-based approaches determine the number of documents to retrieve from a local database based on past retrieval experiences with the database. Several learning-based algorithms in [10] are based on the use of training queries.
6. Weighted allocation and learning-based approaches are heuristic in nature and they do not guarantee that all globally most similar documents will be retrieved from each local search engine.

III. OUR METHOD

To help ordinary users find desired data from the Web, many search engines have been created. Each search engine has a text database that is defined by the set of documents that can be searched by the search engine. In this paper, search engine and database will be used interchangeably. Usually, an inverted file index for all documents in the database is created and stored in the search engine. For each term which can represent a significant word or a combination of several (usually adjacent) significant words, this index can identify the documents that contain the term quickly.

Frequently, the information needed by a user is stored in multiple databases. As an example, consider the case when a user wants to find research papers in some subject area. It is likely that the desired papers are scattered in a number of publishers' databases. Substantial effort would be needed for the user to search each database and identify useful papers from the retrieved papers. A solution to this problem is to implement a metasearch engine on top of many local search engines. A metasearch engine is a system that supports unified access to multiple existing search engines. It does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of its underlying search engines to provide better service. When a metasearch engine receives a user query, it first passes the query to the appropriate local search engines, and then collects the results from its local search engines. With such a metasearch engine, only one query is needed from the above user to invoke multiple search engines.

A closer examination of the metasearch approach reveals the following problems.

1. If the number of local search engines in a metasearch engine is large, then, it is likely that for a given query, only a small percentage of all search engines may contain sufficiently useful documents to the query. In order to avoid or reduce the possibility of invoking useless search engines for a query, we should first identify those search engines that are most likely to provide useful results to the query and then pass the query to only the identified search engines .The problem of identifying potentially useful databases to search is known as the database selection problem.
2. If a user only wants the n most similar documents across all local databases, for some positive integer n, then the n documents to be retrieved from the identified databases need to be carefully specified and retrieved. This is the document selection problem.

Both the problems are described in figure .2.

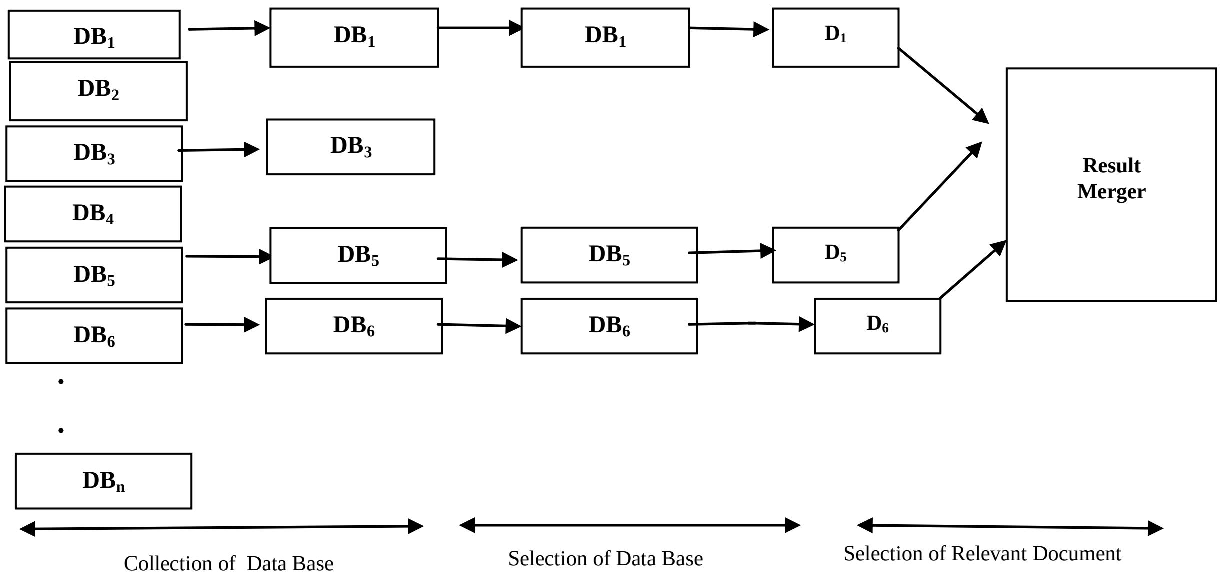


Figure 2: Data base and document selection process

The approach that we propose to retrieve the n most relevant documents across multiple databases for a given query consists of the following two steps:

1. By using SYNCH algorithm first we collect those databases from number of databases which contain our query 'q' and then find the score of each database and then select most appropriate database for the given query 'q'.
2. After databases selection by using MIN algorithms we retrieve 'n' most relevant documents from the selected databases and deleted those document which is similar to the document of other databases.

We want to select those databases from number of databases which contain our query 'q'. For this we proposed an SYNCH Algorithm**.** The Basic idea of this algorithm is that we test databases in the order DB_1, DB_2, DB_3, DB_4, B_5,………, DB_M ,until we get the databases which contain the query 'q'. This algorithm works as follows:

1. Test each database with its documents stored in it. If any document of database contains the query 'q' at least one time then we collect that database.
2. Then we find the score of each database and select those databases which score is maximum and then again find the score of other databases until these databases contain all the document which contain the query 'q'.
3. If all the documents of database does not contains the query 'q' then that database will not be selected.

SYNCH ALGORITHMS FOR DATABASE SELECTION

```
1. Let a given query q
2. list DB[]
3. int score
4. int i
5. int M   // Total number of databases
6. DB_select(DB,q)
7. {
8. for(i=1toM)
9. {
10. if(q=doc_$DB_i$) // query q is find in at least one document)
11. then
12. Collect_DB[]=$DB_i$
13.  break
14. }
15. }
16.  SelectDB(W,n,M,collect_DB)
17. {
18. for(i=1 to M)
19. {
20. find score by  using given formula
21. if (score=MAX)
22. then select $DB_i$
23. else not select DB
24. }
25. }
```

After database selection we retrieve documents from the databases in the order DB_1, DB_2, DB_3, DB_4, $DB_{5\ldots}$ DB_M, until 'n' most relevant documents contained in the selected databases are obtained. For this we proposed an MIN algorithm to retrieve documents from the selected databases. This algorithm works as follows:

1. We search all the selected databases in the order DB_1, DB_2, DB_3, DB_4, DB_5… DB_SWe select only those documents from each database in which the query 'q' occurs.
2. Maintain a list of selected document from each selected database if the new document is not in the list then insert it in to the document list.
3. Rank all the selected documents according to the no. of occurrence of query 'q' in descending order.
4. Return the top 'n' most relevant documents from the ranked list of documents for any positive integer 'n'.

MIN ALGORITHMS FOR DOCUMENT SELECTION

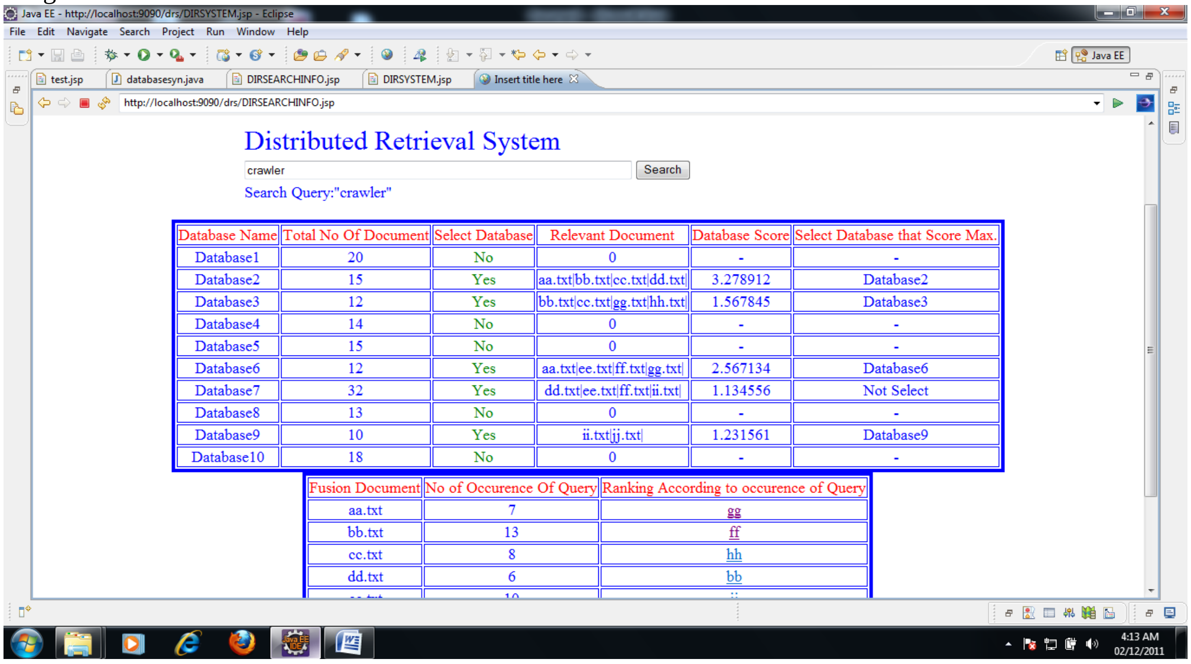
```
1. DOC_Select(q,select$DB_i$)
2. List_Select_Doc[]
3. int S     // total number of selected database
4. for(i=1 to S) //
5. {
6. for(j = 1to n)
7. {
8. if( $doc_j$ = q)
9. if( $doc_j$ != List_select_Doc[i])
10. List_select_Doc[i]= $doc_j$
11. else
12. delete select$DB_i$ from List_Select$DB_i$
13. }
14. }
```

IV. EXPERIMENTAL EVALUATION

Here we compare previous high-correlation method and Subrange-combined-Term method with our SYNCH and MIN algorithms. Here, we compare the performance of the following estimation methods in retrieving the n most relevant documents for n = 5, 10 from the 15 databases for long and Short queries.

1. The high-correlation method does not provide any detail on how a cutoff in database selection is chosen nor which documents are picked from each chosen database.
2. The previous Subrange-combined-term method is useful only for Short queries but it not Efficient for long queries.
3. Our SYNCH algorithm gives the cut off value while selecting the databases and then also find the score of each selected databases and then select most appropriate databases which satisfy the user query ’q’. Thus overhead incurred in processing the databases that are not related to query is minimized.
4. Our MIN algorithm selects the relevant documents from selected databases and merges these documents by using data fusion then all the documents of all selected databases have been ranked. That gives more correct results in comparison with the high-correlation method and Subrange combined-Term method which retrieve documents from the databases, after the databases have been ranked.

Our SYNCH and MIN algorithms were implemented in Advance Java (Servlet) using Eclipse tool. The snapshots of our work are given below.


Java EE - http://localhost:9090/drs/DIRSYSTEM.jsp - Eclipse
File Edit Navigate Search Project Run Window Help
Java EE
test.jsp
databasesyn.java
DIRSEARCHINFO.jsp
DIRSYSTEM.jsp
Insert title here
http://localhost:9090/drs/DIRSEARCHINFO.jsp
4:13 AM
02/12/2011

Distributed Retrieval System

crawler Search

Search Query:"crawler"

Database Name	Total No Of Document	Select Database	Relevant Document	Database Score	Select Database that Score Max.
Database1	20	No	0	-	-
Database2	15	Yes	aa.txt|bb.txt|cc.txt|dd.txt|	3.278912	Database2
Database3	12	Yes	bb.txt|cc.txt|gg.txt|hh.txt|	1.567845	Database3
Database4	14	No	0	-	-
Database5	15	No	0	-	-
Database6	12	Yes	aa.txt|ee.txt|ff.txt|gg.txt|	2.567134	Database6
Database7	32	Yes	dd.txt|ee.txt|ff.txt|ii.txt|	1.134556	Not Select
Database8	13	No	0	-	-
Database9	10	Yes	ii.txt|jj.txt|	1.231561	Database9
Database10	18	No	0	-	-

Fusion Document	No of Occurence Of Query	Ranking According to occurence of Query
aa.txt	7	gg
bb.txt	13	ff
cc.txt	8	hh
dd.txt	6	bb

Figure3: Input pages for query ‘q’ and Results

V. CONCLUSION

The previous database selection and document selection algorithms are based on document term frequency statistics. In this paper we have introduce a new database selection and document selection algorithm for distributed information retrieval environment to collect a databases among various databases and then select most appropriate databases to perform the user query which provide the better result .Our SYNCH algorithm for selecting those databases from number of databases which contain given query ‘q’ and then select most appropriate database which gives the most relevant result and an algorithm MIN to return the top ‘n’ most relevant documents with respect to a given query from a collection of selected databases for any positive integer ‘n’. The future extension of our database selection algorithms in which for finding the score of databases we assume the weighting parameter so it is investigated and explore in future. Our estimation methods are based upon established statistical theory and general database representation framework. Our experimental results indicate that these methods can yield substantial improvements over existing techniques.

REFERENCES

[1] Clement Yu,King-Lup Liu ,weiyi Meng . A. Methodology to Retrieve Documents From Multiple Databases. *IEEE TRANSACTIONS KNOWLEDGE AND DATA ENGINEERING* vol 14 No.6 December 2010.

[2] C. Baumgarten. A Probabilistic Model for Distributed Information Retrieval. *ACM SIGIR Conference*, Philadelphia, 1997.

[3] D. Dreilinger, and A. Howe. Experiences with Selecting Search Engines Using Metasearch. ACM TOIS, 15(3), July 2006.

[4] David Hawking and Paul Thistlewaite. Methods for Information Server Selection. *ACM Transactions on Information Systems*, 17(1):40–76, 2008.

[5] E. Selberg, and O. Etzioni. Multi-Service Search and Comparison Using the MetaCrawler. *4th InternationalWorld WideWeb Conference*, December 1995.

[6] G. Piatetsky-Shapiro and W. J. Frawley. Knowledge Discovery in Databases. AAAI/MIT Press, 1991.

[7] L. Gravano, and H. Garcia-Molina. Merging Ranks from Heterogeneous Internet sources. *International Conferences on Very Large Data Bases*, 1997.

[8] L. Gravano and H. Garcia–Molina. Generalizing GlOSS to Vector–Space Databases and Broker Hierarchies. *In Proceedings of the 21st VLDB Conference, Zurich, Switzerland*, 1995.

[9] L. Gravano, H. Garcia–Molina, and A. Tomasic. Precision and Recall of GlOSS Estimators for Database Discovery. *Stanford University Technical Note Number STAN*-CS-*TN-94-10,* 1994.

[10] N. Craswell, P. Bailey, and D. Hawking. Server Selection on theWorldWideWeb. In Proceedings of the *Fifth ACM Conference on Digital Libraries*, pages 37–46, 2000.

[11] S. Brin and L. Page. The Anatomy of a Large–Scale Hypertextual Web Search Engine. In Proceedings of the WWW7 conference / Computer Networks, volume 1–7, pages 107–117, April 1998.

[12] W. Meng, K.-L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe, “Determining Text Databases to Search in the Internet,” Proc. Int’l Conf. Very Large Data Bases, pp. 14-25, Aug. 1998.

[13] W. Meng, K.-L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe, “Determining Text Databases to Search in the Internet,” Proc. Int’l Conf. Very Large Data Bases, pp. 14-25, Aug. 1998.

[14] Xu, Jinxi, and W.B. Croft. Effective Retrieval with Disributed Collections. In Proceedings of SIGIR98 conference, Melbourne, Australia, August 2008.